

**CSE 1320 Project Documentation**

[**PROJECT**](https://www.studocu.com/in/course/ballari-institute-of-technology-and-management/network-theory/5842786?utm_campaign=shared-document&utm_source=studocu-document&utm_medium=social_sharing&utm_content=c-programming-full-project-documentation) **NAME**

**--Lifeline Donors--**

## **Students names, surnames, IDs:**

## Sujit Shrestha (1002236419)

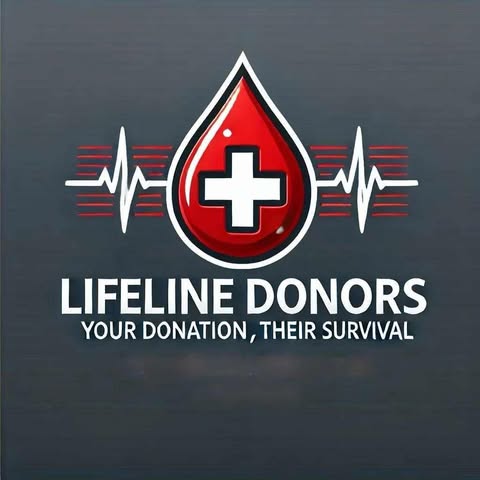
## 2. Siya Darshani Singh Thakuri (1002208756)

## 3. Hady Mourad (1002257563)

## 4. Aashish Neupane (1002032282)

**Mentor: Marika Apostolova**

**PROJECT LOGO**

[](https://github.com/SujitShres1ha/CSE/blob/master/logo.jpg)

**"A donation today, a life saved tomorrow"**

**Intermediate** **programming** **CSE 1320**

**Student** **declaration:**

*We declare* *that:*

* *We* *understand* *what* *is* *meant* *by* *plagiarism*
* *The* *implication* *of* *plagiarism* *has* *been* *explained* *to* *me* *by* *our* *professor*
* *This* *assignment* *is* *all* *team* *own* *work* *and* *we* *have* *acknowledged* *any* *use* *of* *the* *published* *and* *unpublished* *works* *of* *other* *people.*

**1 Student** name, surname, ID and **signature:** Sujit Shrestha ,1002236419

**2 Student** name, surname, ID and **signature:** Siya Darshani Singh Thakuri, 1002208756

**3 Student** name, surname, ID and **signature:** Aashish Neupane, 1002032282

**4 Student** name, surname, ID and **signature:** Hady Mourad, 1002257563

**Date:04/19/2025...........**

|  |  |  |
| --- | --- | --- |
|  | **Total** **number** **of** **pages** **including** **this** **cover** **page** | 6 |
| **Class** **Code** **/** **Group** | CSE 1320 | |
| **Lecturer’s** **Name** | MARIKA APOSTOLOVA | |

**Table of Contents:**

1. Project Introduction ...........................................................................1

2. Project Description ........................................................................... 1

3. Project Architecture .......................................................................... 1

4. Programming Concepts Used ........................................................... 1

5. Code Description .............................................................................. 2

1.Donor Registration with Auto-ID Generation ................................ 2

2.Appointment Scheduling and Slot Selection .................................. 2

3. Donor Information Update ............................................................. 3

4. Pointer-Based Data Handling ......................................................... 3

6. System Testing ................................................................................. 4

7. Group Members & Responsibilities ..................................................5

8. Conclusion and Future Works .......................................................... 6

**1**

**1.Project Introduction:**

Lifeline Donors is a terminal-based blood donation ecosystem that connects donors with recipients through a secure database. This C application implements core healthcare management features while prioritizing accessibility in low-resource environments. The system currently supports donor registration. with medical history screening and lays the foundation for future recipient matching capabilities.

**2.Project Description:**

This system aims to streamline the workflow for both donors and blood center administrators by providing:

**-** Automated Donor Registration: Ensures eligibility and generates unique IDs.

-Appointment Scheduling and Management: Prevents double-booking and displays available slots in real-time.

-Donor Record Maintenance: Allows for safe updates and persistent storage of donor details. All donor and appointment data are stored in plain text files (records.txt and appointments.txt), making the system lightweight and easy to deploy in resource-constrained environments

1. **Project Architecture:**

The system is organized into distinct modules for clarity and maintainability:

├── main.c # Main program loop and user interface

├── customer\_utils.h/c # Donor ID generation, info display, and updating

├── appointments.h/c # Appointment scheduling, changing, and slot management

├── records.txt # Persistent donor records

├── appointments.txt # Persistent appointment records

└── test.c # Basic test routines for core functions

**Figure 1: System architecture**

Data Flow: Users interact with the menu in main.c. Donor operations are handled by customer\_utils. Appointment logic is contained in appointments. All persistent data is read from and written to text files.

1. **Programming Concepts Used:**

* Modular Programming: Functions are organized into header/source files for separation of concerns.
* File I/O: Donor and appointment data are managed using standard file operations (fopen, fprintf, fgets, etc.).
* Structs and Enums: Used for representing donors, appointments, genders, and blood groups.

**2**

* Input Validation: Ensures only eligible donors can register and only valid data is accepted (e.g., age, blood group, gender).
* Error Handling: User feedback is given for invalid inputs or file access errors.
* ANSI Escape Codes: Used for colorful, readable terminal output.
* Testing: Simple test routines provided in test.c.

.

1. **Code Description**
2. **Donor Registration with Auto-ID Generation**

When a user registers, the system generates a unique 4-digit ID and validates all inputs:

int generateCustomerId ()

{

srand(time(NULL));

int size = 0;

char \*\*ids = getExistingCustomerIds (&amp; size);

int customer\_id;

while (1)

{

customer\_id = (rand() % (9999 - 1000 + 1)) + 1000;

if (!isDuplicate(ids, customer\_id, size)) break;

}

for (int i = 0 ; i &lt ; size; i++) free(ids[i]);

free(ids);

return customer\_id;

}

Ensures each donor receives a unique ID and prevents duplicates.

1. **Appointment Scheduling and Slot Selection**

Donors can schedule appointments at their preferred center, with real-time slot availability:

void schedule\_appointment (int customer\_id)

{

// ... (validation and center selection)

char \*appointment = showAppointments(appointment\_center);

if (appointment == NULL) return;

show progress ();

**3**

fprintf (APPOINTMENTS, "\n%d %s %s", customer\_id, appointment\_center, appointment);

fclose(APPOINTMENTS);

print\_success\_message ("Appointment scheduled successfully at center");

free(appointment);

}

Prevents double-booking and allows users to pick from open slots.

**3. Donor Information Update**

Donors can safely update their details. The system ensures atomic updates using a temporary file:

void update\_customer\_info(int customer\_id)

{

// ... (find and display current info)

// Prompt for new values, validate, and update

fprintf(TEMP, "%d %s %d %s %s\n", id, name, age, gender, blood group);

// ... (finalize update)

fclose(RECORD);

fclose(TEMP);

remove("records.txt");

rename ("temp\_appointments.txt", "records.txt");

}

Ensures data consistency and prevents accidental data loss during updates.

**4.Pointer-Based Data Handling**

Double Pointer ID Validation

char \*\*getExistingCustomerIds (int \*size)

{

char \*\*ids = malloc(...);

// Pointer-to-pointer while (fgets(...))

{

ids[i] = malloc(...);

// Nested allocation

}

return ids;

}

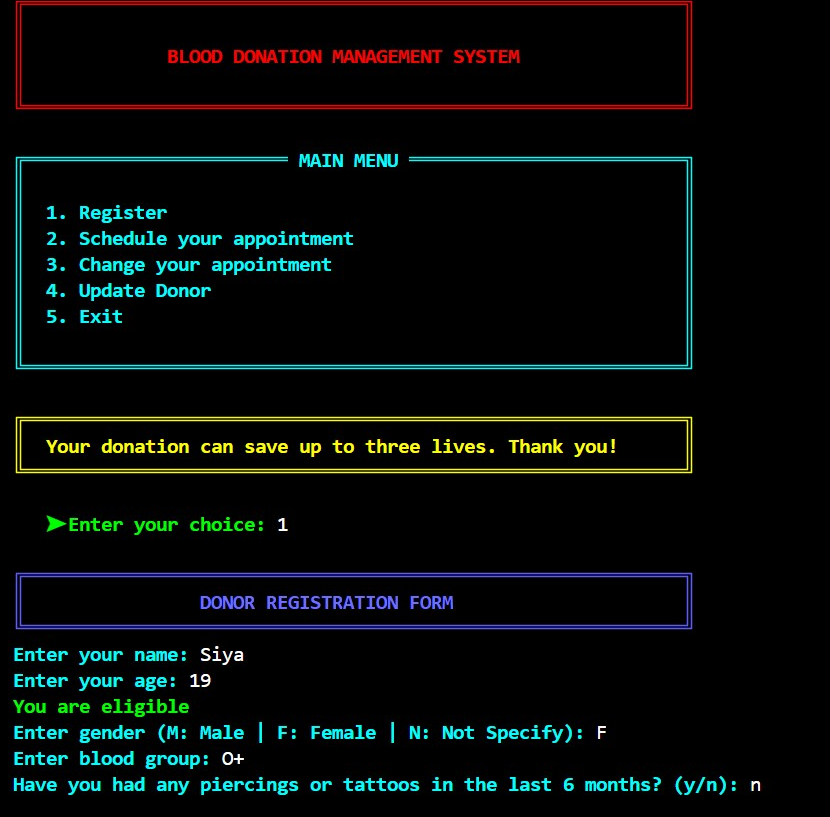
**4**

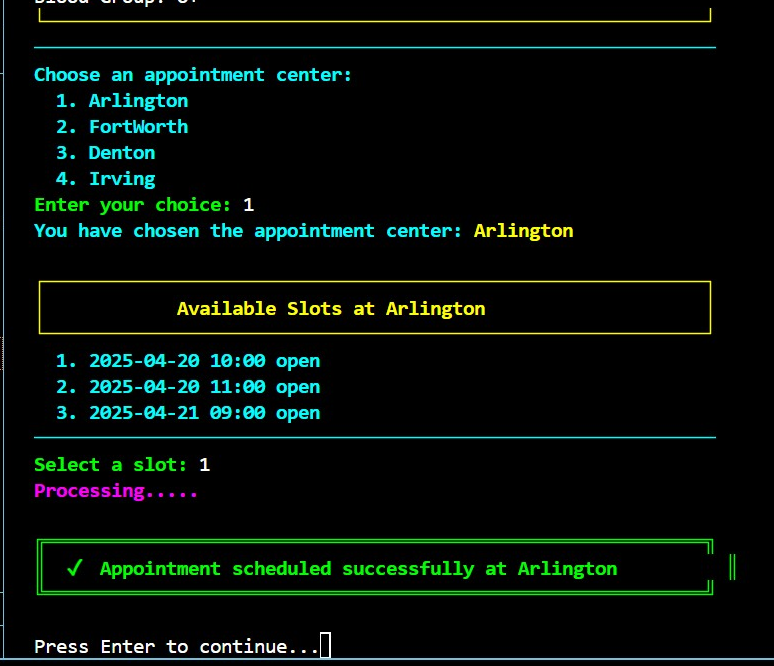
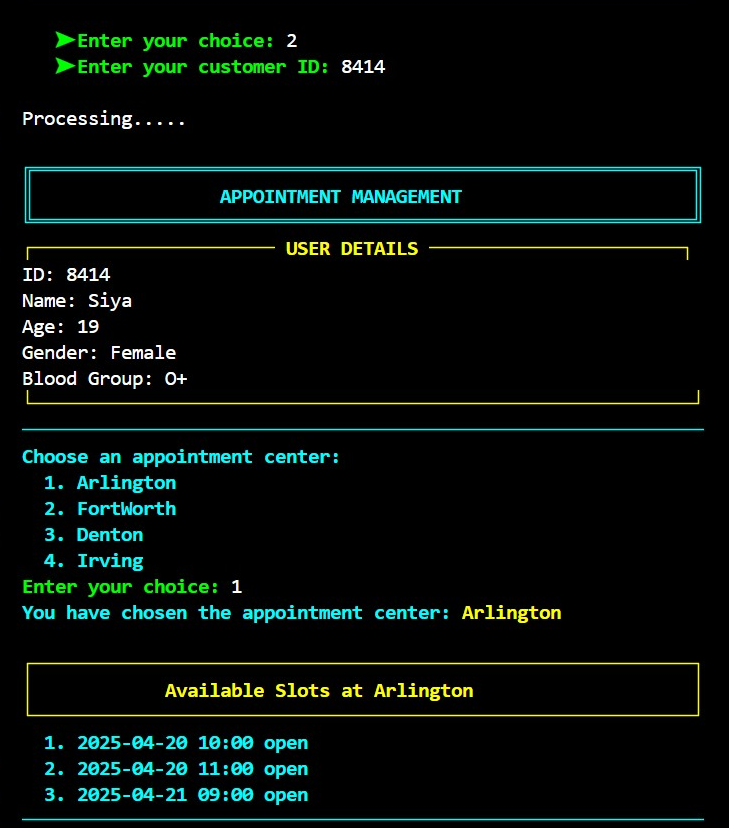
**3.System Testing:**

**Compilation**

gcc main.c customer\_utils.c appointments.c -o lifesaver

./lifesaver

****



**5**

**4.Group members:**

**Member 1(Sujit)**

Project Lead and Main Developer

-Designed the system architecture

-Developed main.c user interface

-Integrated all modules for a seamless flow also troubleshooting the coding syntax.

-Summarized programming concepts and future work suggestions.

**Member 2 (Hady Mourad)**

-Created and tested (customer\_utils.c/h).

-Implemented donor registration, ID generation and data validation logic.

-Helped to debug critical functions.

**Member 3(Aashish)**

-Developed the appointments.c/h module

-Built functions for appointment scheduling, slot management and double- booking prevention.

-Validated input logic and base cases.

**6**

**Member 4(Siya)**

File Handling and Data Persistence

-Managed records.txt and appointments.txt I/O

-Ensured safe updates using temporary files and helped with data consistency and backup.

-Created and executed test cases in test.c

-Drafted a Project report.

**5.Conclusion and future works:**

This project demonstrates how classic C programming techniques can be applied to build a reliable, user-friendly system for managing blood donations. The modular structure, robust validation, and clear terminal interface make it suitable for small clinics and educational use.

**Future Works:**

Network Integration: Add web API support for online donor registration and appointment booking.

Graphical User Interface: Develop a GUI (using GTK+ or Qt) for improved usability.

Advanced Reporting: Implement analytics for donor retention and blood inventory forecasting.

**6.Contribution Guidelines:**

We welcome community contributions through:

-GitHub Issues (bug reports).

-Pull Requests (feature additions).

-Documentation improvements.

**LINK-** [**https://github.com/SujitShres1ha/CSE**](https://github.com/SujitShres1ha/CSE)

**"Every blood drop is a lifeline - handle with care."**

**--Thankyou--**